

# Dynamic Key-Value Memory Network for Knowledge Tracing

Jiani Zhang<sup>1</sup>, Xingjian Shi<sup>2</sup>, Irwin King<sup>1</sup>, Dit-Yan Yeung<sup>2</sup>

<sup>1</sup>{jnzhang, king}@cse.cuhk.edu.hk

Department of Computer Science and Engineering

The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

<sup>2</sup>{xshiab, dyyeung}@cse.ust.hk

Department of Computer Science and Engineering

Hong Kong University of Science and Technology, Kowloon, Hong Kong

## ABSTRACT

The goal of knowledge tracing is to model students' mastering levels of underlying knowledge concepts, termed *knowledge state*, based on students' exercise performance data. However, existing methods, such as Bayesian Knowledge Tracing (BKT) or Deep Knowledge Tracing (DKT), either require costly human-labeled concept annotations or fail to exactly pinpoint which concepts a student is good at or unfamiliar with. To solve these problems, in this paper we introduce a new model called Dynamic Key-Value Memory Network (DKVMN) that can learn representations using nonlinear transformations and directly output a student's mastering level of each concept. Unlike standard Memory-Augmented Neural Networks (MANNs) that facilitate a single memory matrix or two static memory matrices, our model has one static matrix called *key* that stores the knowledge concepts and the other dynamic matrix called *value* that stores and updates corresponding concepts' mastery levels. Experiments show that our DKVMN model, which is trained end-to-end, consistently outperforms the state-of-the-art model on a range of knowledge tracing datasets. We also illustrate that the learned DKVMN can automatically discover underlying concepts of the exercises which are typically performed by human annotations, and depict a student's changing knowledge state.

## CCS Concepts

• **Applied computing** → *Computer-assisted instruction; Learning management systems; E-learning;*

## Keywords

Massive Open Online Courses (MOOCs); Knowledge Tracing; Deep Learning; Dynamic Key-Value Memory Network

## 1. INTRODUCTION

With the advent of Massive Open Online Courses (MOOCs) and Intelligent Tutoring Systems (ITSs) such as Coursera and ASSISTments, students can get appropriate guidance and acquire relevant knowledge in the process of solving exercises. When an exercise is posted, a student must apply one or more concepts to solve the exercise. For example, when a student tries to solve an exercise of '1+2', he or she should apply the concept of 'integer addition'; when a student tries to solve '1+2+3.4', he or she should apply the concepts of 'integer addition' and 'decimal addition'. The probability that the student can answer the exercise correctly is based on the student's *knowledge state*, which stands for the depth and robustness of the underlying concepts the student has mastered.

The goal of *knowledge tracing* (KT) is to trace students' knowledge state based on their exercise performance. KT is an essential task in online learning platforms. For tutors, they can give proper hints and tailor the sequence of practice exercises based on a student's strengths and deficiencies. Study materials which are predicted to be too easy or too hard can be skipped or delayed. For students, they can track their learning progress and devote more energy to less-familiar concepts.

Though effectively modeling students' knowledge has a high educational impact, it is inherently difficult to represent human learning process by numerical simulations. Learning does not happen all at once. It builds upon and is shaped by previous related knowledge in human memory [8]. Therefore, KT is formulated as a supervised sequence learning problem: given a student's past exercise interactions  $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{t-1}\}$ , predict the probability that the student will answer a new exercise correctly, i.e.,  $p(r_t = 1 | q_t, \mathcal{X})$ . Input  $\mathbf{x}_t = (q_t, r_t)$  is a tuple containing the exercise  $q_t$  the student attempts at the timestamp  $t$  and the correctness of the student's answer  $r_t$ . Usually, we model  $\mathcal{X}$  as the observation variables and a student's knowledge state  $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{t-1}\}$  of  $N$  underlying concepts  $\mathcal{C} = \{c^1, c^2, \dots, c^N\}$  as a hidden process. Existing methods model a student's knowledge state either in a concept-specific way or in one summarized hidden state, as shown in Figure 1.

In Bayesian Knowledge Tracing (BKT) [4], a student's *knowledge state* is analyzed into different concept states and

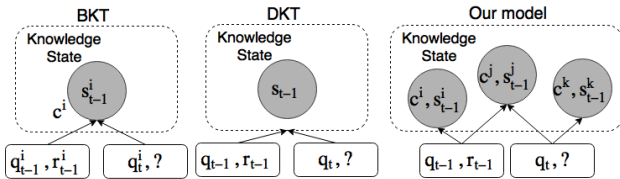


Figure 1: The model differences between BKT, DKT and our model. BKT is concept-specific. DKT uses a summarized hidden state to model knowledge state. Our model maintains *concept state* for each concept simultaneously. And all concept states constitute a student’s *knowledge state*.

BKT models each concept state separately. Then BKT assumes the concept state as a binary latent variable, *known* and *unknown*, and uses Hidden Markov Model (HMM) to update the posterior distribution of the binary concept state. Therefore, BKT cannot catch the relationship between different concepts. Moreover, to keep the Bayesian inference tractable, BKT uses discrete random variables and simple transition models to describe the evolvement of each concept state. As a result, although BKT can output the student’s mastering level of some predefined concepts, it lacks the ability to extract new concepts and to model complex concept state transitions.

Besides solving the problem from the Bayesian perspective, a deep learning method named Deep Knowledge Tracing (DKT) [24] exploits a variant of Recurrent Neural Networks (RNNs) called Long Short-Term Memory (LSTM) [9]. LSTM assumes a high dimensional and continuous representation of the underlying knowledge state  $\mathcal{S}$ . The non-linear input-to-state and state-to-state transitions of DKT have stronger representational power than BKT. No human-labeled annotation is required. However, DKT summarizes a student’s knowledge state of all concepts in one hidden state, which makes it difficult to trace a student’s mastering level of a certain concept and to pinpoint which concepts a student is good at or unfamiliar with.

In this paper, we introduce a new model called Dynamic Key-Value Memory Network (DKVMN) that combines the best of two worlds: the ability to learn representations using nonlinear transformations and the ability to trace each concept state. Our DKVMN model can learn the correlation between input exercises and underlying concepts automatically and it maintains a concept state for each concept. At one timestamp, only related concept states will be updated based on the student’s exercise performance. In Figure 1, to answer exercise  $q_{t-1}$ , a student needs to apply concepts  $c^i$  and  $c^j$ . From the student’s answer response  $r_{t-1}$ , we update the corresponding concept states  $s_{t-1}^i$  and  $s_{t-1}^j$  respectively. All concept states at timestamp  $t-1$  constitute a student’s *knowledge state*  $\mathcal{S}_{t-1}$ . Now we can predict  $p(r_t = 1|q_t, \mathcal{S}_{t-1}) = p(r_t = 1|q_t, s_{t-1}^i, s_{t-1}^k)$  based on the current underlying concept states. Moreover, unlike standard Memory-Augmented Neural Networks (MANNs) that facilitate a single memory matrix [6, 31, 25] or two static memory matrices [28, 18], our model has one static matrix called *key* that stores the concept representations and another dynamic matrix called *value* that stores and updates the student’s understanding (concept state) of the corresponding concepts.

When a new exercise comes, the model will compute a cor-

relation weight vector between the exercise and all concepts in the *key*. Then the correlation weight is used to read the student’s concept states, which are in the *value* component, to predict whether the student will answer the exercise correctly. After the student completes the exercise, the model will use the same correlation weight to update the related concept states. All concept states stored in the *value* component is a student’s dynamic knowledge state.

We summarize our main contributions as follows:

1. We exploit the utility of memory-augmented neural networks to better simulate students’ learning process.
2. We propose a novel Dynamic Key-Value Memory Network with one static *key* matrix and one dynamic *value* matrix.
3. Our model can automatically discover concepts, which is typically performed by human experts and depict students’ changing knowledge state.
4. Our end-to-end trainable model consistently outperforms BKT and DKT on four datasets.

## 2. RELATED WORKS

### 2.1 Knowledge Tracing

Knowledge Tracing (KT) has increasingly received attention in the computer science domain in the last two decades. The KT task evaluates a student’s knowledge state just based on the correctness or incorrectness  $r_t$  of a student’s answers in the process of solving exercises  $q_t$ , where  $q_t$  is an exercise tag and  $r_t \in \{0, 1\}$  is a binary response (1 is correct and 0 is incorrect). No secondary data is incorporated [11].

The dominant method of KT was Bayesian Knowledge Tracing (BKT), which was first introduced in 1994 [4]. For each underlying concept, BKT assumed a two-state learning model, either in the *known* state or in the *unknown* state. Many following variations were raised by integrating personalization study [20, 33], exercise diversity [22] and other information [5, 21] into the Bayesian framework.

Recently, Chris Piech et al. proposed a Deep Knowledge Tracing (DKT) [24] model that exploited the utility of Long Short-Term Memory (LSTM) [9] to trace students’ knowledge. RNNs use hidden states as a kind of summary of the past sequence of inputs and the same parameters were shared over different time steps. Experiments in the paper showed that DKT outperforms previous Bayesian models by a large margin regarding prediction accuracy. This is the first attempt to integrate deep learning models [14, 26], which has achieved great success in other areas like computer vision [15, 13] and natural language processing [3, 17], into knowledge tracing.

### 2.2 Memory-Augmented Neural Networks

Inspired by computer architecture, a particular neural network module called external memory had been proposed to enhance the network’s ability to capture long-term dependencies and to solve algorithmic problems [7]. Memory-Augmented Neural Networks (MANN) had led progress in many areas like question answering [31, 28, 1, 18], natural language transduction [8], algorithm inference [6, 10] and one-shot learning [25, 29].

Typically, the external memory module contains two parts, a memory matrix that stores the information and a controller that communicates with the environment and reads/writes to the memory. The reading and writing operations are achieved through additional attention mechanisms. Most previous works [6, 28, 25] take a similar way to compute the read weight. For an input  $\mathbf{k}_t$ , a cosine similarity or an inner product  $K[\mathbf{k}_t, \mathbf{M}_t(i)]$  of the input and each memory slot is computed, which then goes through a softmax with a positive key strength  $\beta_t$  to obtain a read weight  $\mathbf{w}_t^r$ :  $w_t^r(i) = \text{Softmax}(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(i)])$ , where  $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$ . For the write process, an attention mechanism of focusing both by content and by location is proposed in [6] so as to facilitate all the locations of the memory. Besides, a pure content-based memory writer named LRUA module is raised in [25] that writes the key either to the least recently used memory location or to the most recently used memory location.

Due to the recurrence introduced in the read and write operations, MANN is also a special kind of RNN. However, MANN is different from conventional RNNs like the LSTM model used in DKT in three aspects. Firstly, traditional RNN models use a single hidden state vector to encode the temporal information, which is not compartmentalized enough to accurately remember facts from the past. Instead, MANN uses an external memory matrix to increase the storage capacity [31]. Secondly, the state-to-state transition of traditional RNNs is unstructured and global while MANN uses read and write operations to encourage local state transitions [6]. Thirdly, the number of parameters of traditional RNNs is tied to the size of the hidden states, which is further tied to the memorizing ability [25]. For MANN, increasing the number of memory slots will not increase the parameter number, which is more computationally efficient.

Besides, the recently proposed Key-Value Memory Network in [18] used two static memory matrices, which is not suitable for the KT task. Because static memory cannot simulate a student’s time-varying knowledge state.

### 3. MODEL

In this section, we first introduce the way to exploit existing Memory-Augmented Neural Network (MANN) model to solve the knowledge tracing problem. Then, we show the deficiencies of MANN and go into detail about our own Dynamic Key-Value Memory Network (DKVMN) model.

#### 3.1 Memory-Augmented Neural Network for Knowledge Tracing

To solve the knowledge tracing problem, we treat the inner memory matrix of MANN as the student’s knowledge state. The overall structure of the model is shown in Figure 2a. Here,  $\mathbf{M}_t$  is a  $N \times M$  matrix, where  $N$  is the number of memory locations and  $M$  is the vector size at each location.

At each timestamp  $t$ , the input for MANN is a joint embedding  $\mathbf{s}_t$  of  $(q_t, r_t)$ , where each  $q_t$  comes from a set of  $Q$  distinct exercise tags and  $r_t$  is a binary value indicating whether the student has answered the exercise correctly. The embedding vector  $\mathbf{s}_t$  is used to compute the read weight  $\mathbf{w}_t^r$  and the write weight  $\mathbf{w}_t^w$ . In our implementation, we choose the cosine similarity attention mechanism to compute  $\mathbf{w}_t^r$  and the Least Recently Used Access (LRUA) mechanism [25] to compute  $\mathbf{w}_t^w$ . Details of these two attention mechanisms are

shown in the appendix.

When a student answers exercise that has been stored in the memory and gets the same response,  $\mathbf{s}_t$  will be written to this previously used memory location and when a new exercise arrives or the student gets a different response for the exercise,  $\mathbf{s}_t$  will be written to least recently used memory locations.

In the read process, the read content  $\mathbf{r}_t$  is obtained by the weighted sum of all memory slots with the read weight  $\mathbf{w}_t^r$ :

$$\mathbf{r}_t = \sum_{i=1}^N w_t^r(i) \mathbf{M}_t(i). \quad (1)$$

The output  $\mathbf{p}_t \in \mathbb{R}^Q$ , which is computed from  $\mathbf{r}_t$ , indicates the probability that the student can answer each exercise correctly in the next timestamp.

In the write process, we first erase unnecessary contents in the memory using the erase signal  $\mathbf{e}_t$  and the write weight  $\mathbf{w}_t^r$ , and then add  $\mathbf{s}_t$  into the memory using the add signal  $\mathbf{a}_t$  [6]. For more details, see Section 3.2.3.

MANN can use  $N$  memory slots to encode the student’s knowledge state and has a larger capacity than LSTM which only encodes it in a single hidden state.

#### 3.2 Dynamic Key-Value Memory Network

Despite being more powerful in storing students’ past performance than LSTM, MANN still has deficiencies when applied to the knowledge tracing task. In MANN, the content we read lies in the same space as the content we write. However, for tasks like knowledge tracing, the input and the prediction, which are the exercise the student receives and the correctness of the student’s answer, have different types. This makes it difficult to embed them jointly. Also, MANN does not explicitly model the exercises’ inner concepts. Knowledge state of a particular concept is dispersed and cannot be traced.

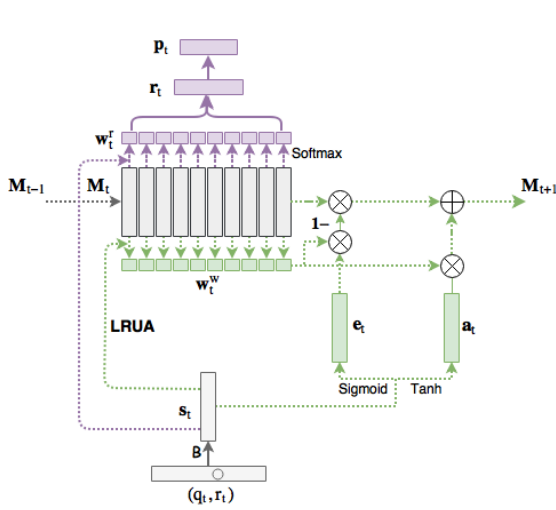
To solve these problems, our Dynamic Key-Value Memory Network (DKVMN) uses key-value pairs rather than a single matrix for the memory structure. Instead of attending, reading and writing to the same memory matrix in MANN, our DKVMN attends input to the *key* component which is fixed all the time, and reads and writes to the corresponding *value* component.

Unlike MANN, at each timestamp, DKVMN takes a discrete exercise tag  $q_t$ , outputs the probability of response  $p(r_t|q_t)$ , and then updates the memory with exercise and response tuple  $(q_t, r_t)$ . Here,  $q_t$  comes from a set with  $Q$  distinct exercise tags and  $r_t$  is a binary value. We further assume there are  $N$  latent concepts  $\{c^1, c^2, \dots, c^N\}$  underlying the exercises. These concepts are stored in the *key* matrix  $\mathbf{M}^k$  (of size  $N \times d_k$ ) and the student’s mastery levels of each concept, i.e., concept states  $\{\mathbf{s}_t^1, \mathbf{s}_t^2, \dots, \mathbf{s}_t^N\}$  are stored in the *value* matrix  $\mathbf{M}_t^v$  (of size  $N \times d_v$ ), which changes over time.

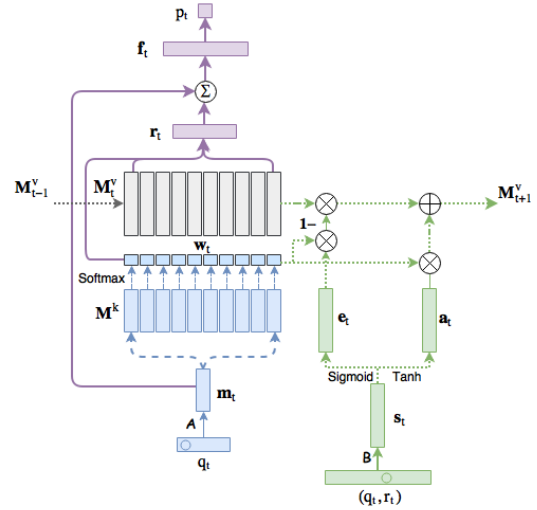
DKVMN traces a student’s knowledge by reading and writing to the *value* matrix using correlation weight computed from the input exercise and the *key* matrix. We will elaborate the model details in the following sections.

##### 3.2.1 Correlation Weight

The input exercise  $q_t$  is first multiplied by an embedding matrix  $\mathbf{A}$  (of size  $Q \times d_k$ ) to get a continuous embedding vector  $\mathbf{m}_t$  of dimension  $d_k$ . The correlation weight is fur-



(a) Architecture for Memory-Augmented Neural Network.



(b) Architecture for Dynamic Key-Value Memory Network.

Figure 2: In both architecture, we only draw the model at the timestamp  $t$ , where the purple components describe the read process and the green components describe the write process. (Best viewed in color.)

ther computed by taking the softmax activation of the inner product between  $\mathbf{m}_t$  and each *key* slot  $\mathbf{M}^k(i)$ :

$$w_t(i) = \text{Softmax}(\mathbf{m}_t^T \mathbf{M}^k(i)), \quad (2)$$

where  $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$  and is differentiable. Both the read and write process will use this weight, which representing the correlation between exercise and each latent concept.

### 3.2.2 Read process

When an exercise  $q_t$  comes, the read content  $\mathbf{r}_t$  is retrieved by the weighted sum of all memory slots in the *value* matrix using  $\mathbf{w}_t$ :

$$\mathbf{r}_t = \sum_{i=1}^N w_t(i) \mathbf{M}_t^v(i). \quad (3)$$

The calculated read content  $\mathbf{r}_t$  is treated as a summary of the student's mastery level of this exercise. Because each exercise has its own difficulty, we concatenate the read content  $\mathbf{r}_t$  and the input exercise embedding  $\mathbf{m}_t$  and then pass it through a fully connected layer with a tanh activation to get a summary vector  $\mathbf{f}_t$ , which contains both the student's mastery level and the prior difficulty of the exercise:

$$\mathbf{f}_t = \text{Tanh}(\mathbf{W}_1^T [\mathbf{r}_t, \mathbf{m}_t] + \mathbf{b}_1), \quad (4)$$

where  $\text{Tanh}(z_i) = (e^{z_i} - e^{-z_i}) / (e^{z_i} + e^{-z_i})$ .

Finally,  $\mathbf{f}_t$  is passed through another fully connected layer with a sigmoid activation to predict the student's performance:

$$p_t = \text{Sigmoid}(\mathbf{W}_2^T \mathbf{f}_t + \mathbf{b}_2), \quad (5)$$

where  $\text{Sigmoid}(z_i) = 1 / (1 + e^{-z_i})$  and  $p_t$  is a scalar that represents the probability of answering  $q_t$  correctly.

### 3.2.3 Write process

After the student answers the question  $q_t$ , the model will update the *value* matrix according to the correctness of the

student's answer. A joint embedding of  $(q_t, r_t)$  will be written to the *value* part of the memory with the same correlation weight  $\mathbf{w}_t$  used in the read process.

The tuple  $(q_t, r_t)$  is embedded with an embedding matrix  $\mathbf{B}$  of size  $2Q \times d_v$  to obtain the student's *knowledge growth*  $\mathbf{s}_t$  after working on this exercise. When writing the student's *knowledge growth* into the *value* component, we erase the memory followed by adding new information [6], which is inspired by the input and forget gates in LSTMs.

Given a write weight (which is the correlation weight  $\mathbf{w}_t$  in our model), we compute an *erase vector*  $\mathbf{e}_t$  from  $\mathbf{s}_t$ :

$$\mathbf{e}_t = \text{Sigmoid}(\mathbf{E}^T \mathbf{s}_t + \mathbf{b}_e), \quad (6)$$

where the transformation matrix  $\mathbf{E}$  is of shape  $d_v \times d_v$ ,  $\mathbf{e}_t$  is a column vector with  $d_v$  elements all lie in the range  $(0, 1)$ . The memory vectors of *value* component  $\mathbf{M}_{t-1}^v(i)$  from the previous timestamp are modified as follows:

$$\tilde{\mathbf{M}}_t^v(i) = \mathbf{M}_{t-1}^v(i) [\mathbf{1} - w_t(i) \mathbf{e}_t], \quad (7)$$

where  $\mathbf{1}$  is a row-vector of all 1-s. Therefore, the elements of a memory location are reset to zero only if both the weight at the location and the erase element are one. The memory vector is left unchanged if either the weight or the erase signal is zero.

After erasing, we use a length  $d_v$  *add vector*  $\mathbf{a}_t$  to update each memory slot:

$$\mathbf{a}_t = \text{Tanh}(\mathbf{D}^T \mathbf{s}_t + \mathbf{b}_a)^T, \quad (8)$$

where transformation matrix  $\mathbf{D}$  is of shape  $d_v \times d_v$  and  $\mathbf{a}_t$  is a row vector. The *value* memory is updated at each time  $t$  by

$$\mathbf{M}_t^v(i) = \tilde{\mathbf{M}}_{t-1}^v(i) + w_t(i) \mathbf{a}_t. \quad (9)$$

This *erase*-followed-by-*add* mechanism allows forgetting and strengthening concept states in a student's learning process.

### 3.2.4 Training

Table 1: Test AUC results for all datasets. BKT is the standard BKT. BKT+ is the best-reported result with BKT variations. DKT is the result using LSTM. MANN is the baseline using a single memory matrix.

Datasets	Overview			Test AUC (%)				
	Students	Exercise Tags	Records	BKT	BKT+	DKT	MANN	DKVMN
Synthetic-5	4,000	50	200,000	62	80	80.3±0.1	81.0±0.1	<b>82.7±0.1</b>
ASSISTments2009	4,151	110	325,637	63	-	74.1±0.2	73.5±0.2	<b>74.7±0.1</b>
ASSISTments2015	19,840	100	683,801	64	-	72.5±0.1	72.3±0.2	<b>72.7±0.1</b>
Statics2011	333	1,223	189,297	73	75	80.2±0.2	77.6±0.1	<b>82.8±0.1</b>

The overall model architecture is shown in Figure 2b. During training, both embedding matrices  $\mathbf{A}$  and  $\mathbf{B}$ , as well as other parameters and the initial value of  $\mathbf{M}_0^k$  and  $\mathbf{M}_0^v$  are jointly learned by minimizing a standard cross entropy loss between  $p_t$  and the true label  $r_t$ .

$$\mathcal{L} = - \sum_t (r_t \log p_t + (1 - r_t) \log(1 - p_t)). \quad (10)$$

Our DKVMN model is fully differentiable and can be trained efficiently with stochastic gradient descent (see Section 4.2 for more details).

## 4. EXPERIMENTS

We first evaluate the prediction accuracy by comparing our Dynamic Key-Value Memory Network (DKVMN) with other methods on four datasets, one synthetic dataset and three real-world datasets collected from online learning platforms. Then, comparative experiments of different dimensions of states are done on DKVMN and DKT for further model exploration. Finally, we verify our model’s ability to discover concepts automatically and depict students’ knowledge state.

The results of the experiments lead to the following findings:

- DKVMN outperforms the standard MANN and the state-of-the-art method on four datasets.
- DKVMN can produce better results with fewer parameters than DKT.
- DKVMN does not suffer from overfitting, which is a big issue for DKT.
- DKVMN can discover underlying concepts precisely and depict students’ concept states of distinct concepts over time.

We implement the models using MXNet [2]. All experiments are run on a computer with a single NVIDIA K40 GPU.

### 4.1 Datasets

To evaluate performance, we test knowledge tracing models on four datasets: Synthetic-5, ASSISTments2009, ASSISTments2015, and Statics2011.

**Synthetic-5:** This dataset<sup>1</sup> simulates 2000 virtual students answering 50 exercises with the same sequence order both in the training and testing dataset. Each exercise is drawn from one of five hidden concepts and has different

<sup>1</sup>Synthetic-5: <https://github.com/chrispiech/DeepKnowledgeTracing/tree/master/data/synthetic>

difficulties. We have no access to the underlying concept labels in the training process and just use them as the ground truth to evaluate the discovered concept results using our DKVMN model.

**ASSISTments2009:** This dataset was gathered in the year 2009-2010 from the ASSISTments platform. ASSISTments is an online tutoring platform that evaluates students with pre-built problem sets. The ‘skill-builder’ dataset is a large, standard benchmark [11, 24]. Due to duplicated record issues [32], an updated version was released this year and all previous results on the old dataset were no more reliable. The experiments in our paper were conducted using the updated dataset<sup>2</sup>. Records without skill names were discarded in the preprocessing. Thus, the number of records in our experiments is smaller than that in [32]. 4,151 students answered 325,637 exercises along with 110 distinct exercise tags.

**ASSISTments2015:** This dataset<sup>3</sup> is also from ASSISTments, but gathered in the year of 2015. ASSISTments2015 only contains student responses on 100 skill builders with the highest number of student responses. After preprocessing (removing the value of *correct*  $\notin \{0, 1\}$ ), this dataset remains 683,801 effective records from 19,840 students. Each problem set in this dataset has only one associated skill. Although this dataset has the largest number of records, the average records for each student is also the lowest.

**Statics2011:** Statics<sup>4</sup> is from a college-level engineering statics course with 189,297 trials, 333 students and 1,223 exercises tags [27], available from the PSLC DataShop web site [12]. In our experiments, we use a concatenation of problem name and step name as an exercise tag; thus it has the maximum number of exercise tags and also the maximum number of average records per student.

The complete statistic information of all datasets can be found in Table 1.

### 4.2 Implementation Details

The input exercise data is presented to neural networks using ‘one-hot’ input vectors. That is, if there are  $Q$  different exercises in total, the exercise tag  $q_t$  for *key* memory part is a length  $Q$  vector whose entries are all zero except for the  $q_t^{th}$  entry, which is one. Similarly, the combined input  $x_t = (q_t, r_t)$  for the *value* matrix component is a length  $2Q$  vector, where entry  $x_t = q_t + r_t * Q$  is one.

We learn the initial value of both the *key* and the *value*

<sup>2</sup>ASSISTments2009: <https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010>

<sup>3</sup>ASSISTments2015: <https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data>

<sup>4</sup>Statics2011: <https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507>

Table 2: Comparison of DKVMN with DKT on four datasets with a different number of state dimension and memory size. ‘s. dim’, ‘m. size’ and ‘p. num’ represent state dimension, memory size, and the number of parameters. We choose state dimension of 10, 50, 100, and 200 both for DKT and DKVMN. Then for DKVMN, we change memory size for 1, 2, 5, 10, 20, 50, 100 for each state dimension and report the best test AUC with the corresponding memory size. Also, we compare the number of parameters for both models.

Model	Synthetic-5				ASSISTments2009				ASSISTments2015				Statics2011			
	s. dim	m. size	test auc	p. num	s. dim	m. size	test auc	p. num	s. dim	m. size	test auc	p. num	s. dim	m. size	test auc	p. num
DKT	10	-	80.06	2.4K	10	-	73.88	4.3K	10	-	72.40	4.0K	10	-	78.12	39K
	50	-	80.22	28K	50	-	74.11	37K	50	-	<b>72.52</b>	36K	50	-	79.86	205K
	100	-	<b>80.34</b>	96K	100	-	<b>74.12</b>	114K	100	-	72.49	111K	100	-	80.16	449K
	200	-	80.32	352K	200	-	74.10	388K	200	-	72.45	382K	200	-	<b>80.20</b>	1.0M
DKVMN	10	50	82.00	12K	10	50	<b>74.69</b>	17K	10	20	<b>72.68</b>	14K	10	10	82.72	92K
	50	50	82.66	25K	50	10	74.58	30K	50	10	72.66	29K	50	10	<b>82.84</b>	197K
	100	50	<b>82.73</b>	50K	100	20	74.63	61K	100	50	72.64	63K	100	10	82.71	338K
	200	50	82.71	130K	200	10	74.35	148K	200	50	72.53	153K	200	10	82.70	649K

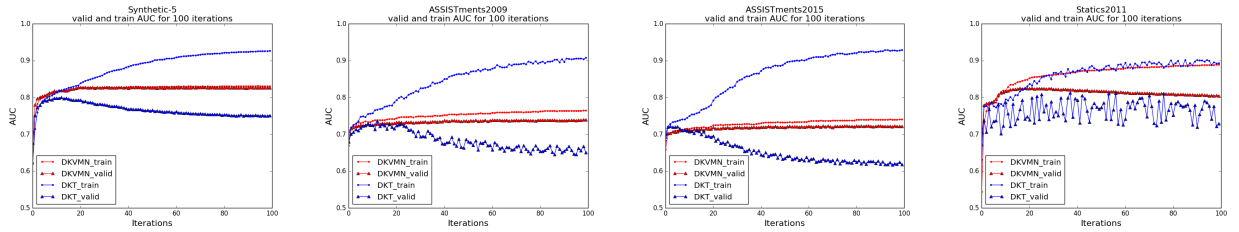


Figure 3: The validation AUC and training AUC of DKVMN and DKT on all datasets. The line in blue stands for the DKT model and the line in red stands for our DKVMN model. The line annotated with small dot is the training AUC and the line with upper triangle is the validation AUC. (Best viewed in color.)

matrix in the training process. Each slot of the *key* memory is the concept embedding and is fixed in the testing process. Meanwhile, the initial value of the *value* memory is the initial state of each concept, which represents the initial difficulty of each concept.

Of all the datasets, 30% of the sequences was held out as a testing set, except for the synthetic dataset where training and testing dataset had the same size. 20% of the training set was split to form a validation set, which was used to select the optimal model architecture, hyperparameters, and perform early-stopping [19]. The parameters were initialized randomly from a Gaussian distribution with zero mean and standard deviation  $\sigma$ . The initial learning rate was case by case, since the number of students, exercise tags and total answers per dataset varied, but the learning rate  $\gamma$  annealed every 20 epochs by  $\gamma/1.5$  until the 100 epoch was reached. We use LSTM for Deep Knowledge Tracing (DKT) in our implementation. The standard MANN is implemented using the cosine similarity reading attention mechanism and the LRUA writing attention mechanism. We used stochastic gradient descent with momentum and norm clipping [23] to train DKT, MANN and our Dynamic Key-Value Memory Network (DKVMN) in all the experiments. We consistently set the momentum to be 0.9 and the norm clipping threshold to be 50.0. Because input sequences are of different lengths, we split all sequences to length 200 (for *synthetic* with length 50) and a null symbol was used to pad short sequence to a fixed size of 200. In all cases, hyperparameters were tuned using the five-fold cross validation. The test AUC was computed using the model with the highest valida-

tion AUC among the 100 epochs. We repeated each training five times with different initializations  $\sigma$  and reported the average test AUC along with the standard deviation.

### 4.3 Student-performance Prediction

We measure the Area Under the Curve (AUC) to evaluate the prediction accuracy on each dataset. An AUC of 50% represents the score achievable by random guess. A higher AUC score accounts for a better prediction performance. Results of the test AUC on all datasets are shown in Table 1.

We compare the DKVMN model with the MANN baseline, the state-of-the-art DKT, the standard Bayesian Knowledge Tracing (BKT) model and, when possible, with optimal variations of BKT (BKT+). An interesting observation is that our implemented LSTM achieves better AUC than those in the original papers [24, 11, 32]. The reason may be that our implementations use norm clipping and early-stopping which improve the overfitting problem of LSTM. The results of BKT are directly obtained from recent work [11, 32].

On the Synthetic-5 dataset, DKVMN model achieves the average test AUC of 82.7%. In our simulation, we treat each exercise as having a distinct skill label. The MANN produces an average AUC of 81.0%. The DKT produces an AUC value of 80.3%, which is better than that of 75% reported in the original paper [24, 11]. The BKT and its variant model achieve the AUC of 62% and 80% [11]. The prediction results of DKVMN from the ASSISTments2009 obtain small improvement over MANN, DKT, and BKT with 74.7% over 73.5%, 74.1%, and 63% [32]. Because we preprocess this dataset different from that in [32], their results are



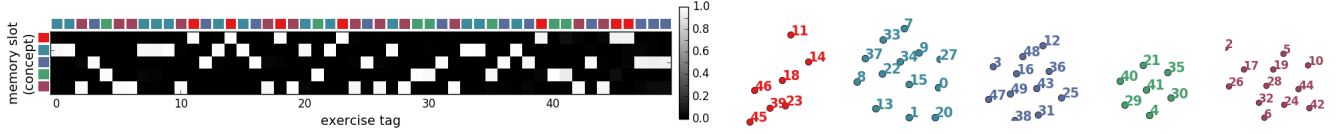


Figure 4: Concept discovery results on the synthetic dataset. In the left heat map, the x-axis represents each exercise and the y-axis represents the correlation weight between the exercise and five latent concepts generated from our DKVMN model. The ground-true concept is labeled on the top of each exercise. In the right exercise clustering graph, each node number represents an exercise. Exercises from the same ground-truth concept are clustered together. (Best viewed in color.)

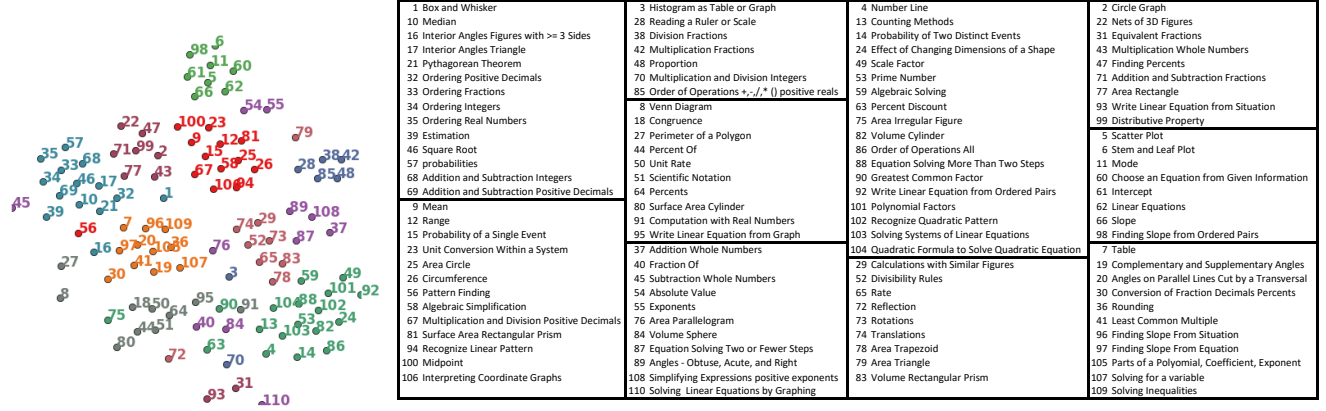


Figure 5: Concept discovery results on the ASSISTments2009 dataset. 110 exercises are clustered into ten concepts. Exercises under the same concept are labeled in the same color in the left picture and also are put in the same block in the right table. (Best viewed in color.)

not comparable. On the ASSISTments2015 dataset, the test AUC of DKVMN is 72.7%, which is better than 72.3% for MANN, 72.5% for DKT (originally is 70% in [32]) and 64% for classic BKT [32]. Turning to Statics2011, which has the maximum number of exercise tags and the minimum number of answers, classical BKT gains the AUC of 73% and BKT cooperating with forgetting, skill discovery and latent abilities obtains an AUC of 75% [11]. Our implemented DKT leads to an AUC of 80.2%, which is better than that of 76% from [11]. The MANN only produces the average AUC of 77.6%. However, our DKVMN model achieves an AUC of 82.8%, outperforms all previous models.

In summary, DKVMN performs better than other methods across all the datasets, in particular on the Statics2011 dataset whose number of distinct exercises is large. This result demonstrates that our DKVMN can model student’s knowledge well when the number of exercises is very large.

DKVMN not only can achieve better prediction accuracy over student exercise performance but also requires much fewer parameters than the DKT model because of the large external memory capacity. Table 2 compares the DKVMN model with the DKT model using LSTM by traversing different hyperparameters. From the table, we find that DKVMN with low state dimensions can achieve better prediction accuracy than DKT with high state dimensions. For instance, on the synthetic-5 dataset, DKT reaches the maximum test AUC of 80.34% when the dimension of states equals to 100 using 96K parameters. Meanwhile, DKVMN can achieve the test AUC of 82.00% only with ten state dimensions using 12K parameters.

Moreover, the DKT model suffers severe overfitting, while

our DKVMN model does not confront with such a problem. As indicated in Figure 3, there is no huge gap between the training AUC and the validation AUC of DKVMN and the validation AUC of DKVMN increases smoothly. However, as the epoch proceeds, the training AUC of DKT increases continuously and the validation AUC of DKT only increases in the first several epochs and begins to decrease.

## 4.4 Concept Discovery

Our DKVMN model has the power to discover underlying patterns or concepts for exercises using the correlation weight  $w$ , which is traditionally annotated by experts. The correlation weight between the exercise and the concept implies the strength of their inner relationship. Compared with the conditional influence approach in [24] which computes the dependencies between exercises and then defines a threshold to cluster the exercises, our model directly assigns exercises to concepts. No pre-defined threshold is required. As a result, our model can discover the concepts of exercises in an end-to-end manner.

Usually, each exercise is associated with a single concept. In this case, we assign the exercise to the concept with the largest correlation weight value. From the experiments, we find that our model can intelligently learn sparse weight among concepts for the exercises and the discovered concepts reveal a compelling result.

On the Synthetic-5 dataset, each exercise was drawn from a concept  $c^k$ , where  $k \in 1..5$ , so we can access the ground-truth concept for all exercises, which is shown on the top x-axis of the heat map in Figure 4. Exercises from the same concept are labeled with squares in the same color. The left heat map in Figure 4 shows the correlation weight between

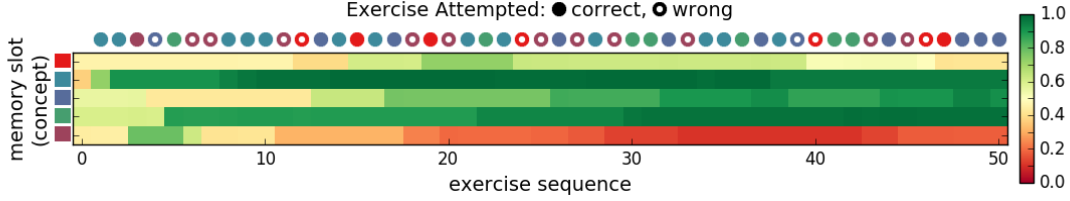


Figure 6: An example of a student’s changing knowledge state on five concepts. Concepts are marked in different colors on the left side. After answering fifty exercises, the student masters the second, third, and forth concepts but fails to understand the fifth concept. (Best viewed in color.)

fifty distinct exercises and five latent concepts (generated from DKVMN when the memory size is five). Each column represents the correlation weight between an exercise and five latent concepts. For each exercise, the weight is sparse where exactly one value approximates to 1 and the others approximate to 0. After clustering each exercise to the concept with the maximum weight value, we get the graph shown in the right part of Figure 4), which reveals a perfect clustering of five latent concepts. The adjusted mutual information [30] of our clustering result and the ground truth is 1.0.

On the ASSISTments2009 dataset, there is no ground-truth concept for each exercise. However, we can obtain the name for each exercise tag, as shown in the right part of Figure 5. Each exercise tag is followed by a name. The resulting cluster graph in Figure 5 is drawn using t-SNE [16] by projecting the multi-dimensional correlation weights to the two-dimensional points. We group all exercises into ten clusters, where the exercises from the same cluster (concept) are labeled in the same color. The clustering graph reveals many reasonable results. Some related exercises are close to each other in the cluster. For example, in the first cluster, 32 *Ordering Positive Decimals*, 33 *Ordering Fractions*, 34 *Ordering Integers* and 35 *Ordering Real Numbers* are clustered together, which exposes the concept of *elementary arithmetic*.

#### 4.5 Knowledge State Depiction

Our DKVMN can also be used to depict students’ changing knowledge state. Depicting the knowledge state, especially each concept state, is helpful for the users on online learning platforms. If students possess their concept states of all concepts, which pinpoint their strengths and weaknesses, they will be more motivated to fill in the learning gaps independently. A student’s changing knowledge state can be obtained in the read process using the following steps:

First, the content in the *value* component is directly used as the read content  $\mathbf{r}_t$  in Eq.(3), which can be accessed by setting the correlation weight  $\mathbf{w}_t$  to be  $[0, \dots, w_i, \dots, 0]$ , where  $w_i$  is equal to 1. Then, we mask the weight of the input content embedding in Eq.(4) to ignore the information of exercises:

$$\mathbf{f}_t = \text{Tanh}([\mathbf{W}_1^r, \mathbf{0}]^T [\mathbf{r}_t, \mathbf{m}_t] + \mathbf{b}_1), \quad (11)$$

where  $\mathbf{W}_1$  is split into two parts  $\mathbf{W}_1^r$  and  $\mathbf{W}_1^m$ , and let  $\mathbf{W}_1^m = \mathbf{0}$ . Finally, we compute the scalar  $p$  as in Eq.(5) to be the predictive mastery level of a concept (concept state).

Figure 6 shows an example of depicting a student’s five changing concept concepts. The first column represents the initial state of each concept before the student answers any

exercise, which differs from concept to concept. Because of our model’s ability to discover concepts for each exercise, each time the student answers an exercise, the concept state of the discovered concept will increase or decrease. For example, when the student answers the first three exercises correctly, concept states of the second and the fifth concepts increase; when the student answers the fourth exercise incorrectly, the concept state of the third concept decreases. After answering fifty exercises, the student is shown to have mastered the second, third, and forth concepts but fail to understand the fifth concept.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a new sequence learning model called Dynamic Key-Value Memory Network (DKVMN) to tackle the knowledge tracing problem, which can be implemented in online learning platforms to improve students’ study efficiency. DKVMN not only outperforms the state-of-the-art Deep Knowledge Tracing (DKT) but also can trace a student’s understanding of each concept over time, which is the main drawback of DKT. Compared with standard Memory-Augmented Neural Networks (MANNs), the key-value pair allows DKVMN to discover underlying concepts for each input exercise and to trace a student’s knowledge state of all concepts.

For future work, we will incorporate content information into the exercise and concept embeddings to further improve the representations. We will also investigate a hierarchical key-value memory network structure which can encode the hierarchical relationship between concepts.

## 6. APPENDIX

### 6.1 Read Attention Mechanism of MANN

For each input key  $\mathbf{k}_t$ , we compute the cosine similarity of the key and memory:

$$K[\mathbf{k}_t, \mathbf{M}_t(i)] = \frac{\mathbf{k}_t \cdot \mathbf{M}_t(i)}{\|\mathbf{k}_t\| \cdot \|\mathbf{M}_t(i)\|}, \quad (12)$$

which is then used to compute the read weight  $\mathbf{w}^r$  through a softmax with a positive key strength  $\beta_t$ :

$$w_t^r(i) = \frac{\exp(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(i)])}{\sum_j \exp(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(j)])}. \quad (13)$$

### 6.2 Write Attention Mechanism of MANN

Least Recently Used Access (LRUA) model [25] writes the keys either to the least used memory location or the most recently used memory location.



First, a usage weight vector  $\mathbf{w}_t^u$  is used to record the usage frequency of all memories. The usage weights are updated at each time-step by decaying the previous usage weights and adding the current reading and writing weights:

$$\mathbf{w}_t^u = \gamma \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w, \quad (14)$$

where  $\gamma$  is a decay parameter. We fix  $\gamma$  to be 0.9 in our implementation.

Then the least-used weight  $\mathbf{w}_t^{lu}$  is defined to record the least-used memories using a notation  $m(\mathbf{v}, n)$ , which denotes  $n^{th}$  smallest element of the vector  $\mathbf{v}$ .

$$w_t^{lu}(i) = \begin{cases} 0 & \text{if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 & \text{if } w_t^u(i) \leq m(\mathbf{w}_t^u, n), \end{cases} \quad (15)$$

where  $n$  is set to equal the number of reads to memory.

Now the write weight  $\mathbf{w}_t^w$  is the convex combination of the previous read weights and previous least-used weights:

$$\mathbf{w}_t^w = \sigma(\alpha) \mathbf{w}_{t-1}^s + (1 - \sigma(\alpha)) \mathbf{w}_{t-1}^{lu}, \quad (16)$$

where  $\sigma(\cdot)$  is a sigmoid function and  $\alpha$  is a scalar gate parameter to interpolate between two weights.

## 7. REFERENCES

- [1] A. Bordes, N. Usunier, S. Chopra, and J. Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.
- [2] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. 2015.
- [3] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [4] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. volume 4, pages 253–278. Springer, 1994.
- [5] R. S. d Baker, A. T. Corbett, and V. Alevan. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *Intelligent Tutoring Systems*, pages 406–415. Springer, 2008.
- [6] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- [7] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016.
- [8] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015.
- [9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [10] A. Joulin and T. Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, pages 190–198, 2015.
- [11] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? In *Proceedings of the 9th International Conference on Educational Data Mining*, 2016.
- [12] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, 43, 2010.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [14] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [15] Y. LeCun, K. Kavukcuoglu, C. Farabet, et al. Convolutional networks and applications in vision. In *ISCAS*, pages 253–256, 2010.
- [16] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.
- [17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [18] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.
- [19] G. B. Orr and K.-R. Müller. *Neural networks: tricks of the trade*. Springer, 2003.
- [20] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.
- [21] Z. A. Pardos and N. T. Heffernan. Using hmms and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. *Proceedings of the KDD Cup 2010 Workshop*, 2010.
- [22] Z. A. Pardos and N. T. Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *User Modeling, Adaption and Personalization*, pages 243–254. Springer, 2011.
- [23] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of The 30rd International Conference on Machine Learning*, 28:1310–1318, 2013.
- [24] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, 2015.
- [25] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. pages 1842–1850, 2016.
- [26] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
- [27] P. Steif and N. Bier. Oli engineering statics - fall 2011. Feb. 2014.
- [28] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end

- memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.
- [29] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. 2016.
- [30] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.
- [31] J. Weston, S. Chopra, and A. Bordes. Memory networks. *International Conference on Learning Representations*, 2015.
- [32] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining*, 2016.
- [33] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *Artificial intelligence in education*, pages 171–180. Springer, 2013.